

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application

Applicant(s): K. Betz et al.
Docket No.: YO999-547
Serial No.: 09/500,208
Filing Date: February 8, 2000
Group: 2157
Examiner: Ramy M. Osman

Title: Methods and Apparatus for Reducing the Number of Server Interactions
In Network-Based Applications Using a Dual-MVC Approach

APPEAL BRIEF

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Applicants (hereinafter referred to as “Appellants”) hereby appeal the final rejection of claims 1-26 of the above-identified application.

REAL PARTY IN INTEREST

The present application is assigned to International Business Machines Corporation, as evidenced by an assignment recorded February 8, 2000 in the U.S. Patent and Trademark Office at Reel 10556, Frame 866. The assignee, International Business Machines Corporation, is the real party in interest.

RELATED APPEALS AND INTERFERENCES

There are no known related appeals or interferences.

STATUS OF CLAIMS

Claims 1, 3, 13, 15, 25 and 26 stand finally rejected under 35 U.S.C. §102(e). Claims 2, 4-12, 14 and 16-24 stand finally rejected under 35 U.S.C. §103(a). Claims 1-26 are appealed.

STATUS OF AMENDMENTS

There has been no amendment filed subsequent to the final rejection.

SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1 recites a method for use in a client/server system of reducing interactions between a client and server in association with an application being accessed by the client at the server, the method comprising the steps of: configuring the server to store a model associated with the application and to execute view-generating and controller logic associated with the application; and configuring the client to store at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic associated with the application; wherein one or more portions of the application are performed at the client without the client having to interact with the server, and further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith.

An illustrative embodiment of the recited method for use in a client/server system (e.g., FIG. 3) of reducing interactions between a client (e.g., web browser 1000 in FIG. 3) and server (e.g., web server 5000 in FIG. 3) in association with an application being accessed by the client at the server (e.g., Specification, page 8, lines 21-27, “a help desk application”), the method comprising the steps of: configuring the server (e.g., web server 5000 in FIG. 3) to store a model associated with the application (e.g., server-side model 5600 in FIG. 3) and to execute view-generating (e.g., server-side view-generating logic 5400 in FIG. 3) and controller logic (e.g., server-side controller logic 5800 in FIG. 3) associated with the application; and configuring the client (e.g., web browser 1000 in FIG. 3) to store at least a subset of the model (e.g., client-side model 1600 in FIG. 3) associated with the application and to execute at least a subset of the view-generating (client-side view-generating logic

1400 in FIG. 3) and controller logic (e.g., client-side controller logic 1800 in FIG. 3) associated with the application; wherein one or more portions of the application are performed at the client without the client having to interact with the server (e.g., Specification, page 10, lines 16-18, “when a subset of the Model is stored on the client, as in accordance with the dual-MVC approach of the present invention, a number of interactions with the server can be eliminated”), and further wherein the client and the server both locally maintain at least a portion of the model (e.g., client-side model 1600 and server-side model 5600 in FIG. 3) and execute the view-generating (e.g., client-side view-generating logic 1400 and server-side view-generating logic 5400 in FIG. 3) and controller logic (e.g., client-side controller logic 1800 and server-side controller logic 5800) associated therewith.

Independent claim 13 recites a network-based system, comprising: a server having at least one processor operative to: (i) store a model associated with an application associated with the server; and (ii) execute view-generating and controller logic associated with the application; and a client, coupled to the server via a network, having at least one processor operative to: (i) store at least a subset of the model associated with the application; and (ii) execute at least a subset of the view-generating and controller logic associated with the application; wherein one or more portions of the application are performed at the client without the client having to interact with the server such that interactions between the client and server are reduced, and further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith.

In an illustrative embodiment of the recited network-based system, comprising: a server (e.g., web server 5000 in FIG. 3) having at least one processor (e.g., processor 9000 in FIG. 10) operative to: (i) store a model (e.g., server-side model 5600 in FIG. 3) associated with an application associated with the server; and (ii) execute view-generating (e.g., server-side view-generating logic 5400 in FIG. 3) and controller logic (server-side controller logic 5800 in FIG. 3) associated with the application; and a client (e.g., web browser 1000 in FIG. 3), coupled to the server (e.g., server 5000 in FIG. 3) via a network (e.g., HTTP network 3000 in FIG. 3), having at least one processor (e.g., processor 9000 in FIG. 10) operative to: (i) store at least a subset of the model (e.g., client-side model 1600 in FIG. 3) associated with the application; and (ii) execute at least a subset of the view-

generating (client-side view-generating logic 1400 in FIG. 3) and controller logic (e.g., client-side controller logic 1800 in FIG. 3) associated with the application; wherein one or more portions of the application are performed at the client without the client having to interact with the server such that interactions between the client and server are reduced (e.g., Specification, page 10, lines 16-18, “when a subset of the Model is stored on the client, as in accordance with the dual-MVC approach of the present invention, a number of interactions with the server can be eliminated”), and further wherein the client and the server both locally maintain at least a portion of the model (e.g., client-side model 1600 and server-side model 5600 in FIG. 3) and execute the view-generating (e.g., client-side view-generating logic 1400 and server-side view-generating logic 5400 in FIG. 3) and controller logic (e.g., client-side controller logic 1800 and server-side controller logic 5800) associated therewith.

Independent claim 25 recites an article of manufacture for use in a client/server system for reducing interactions between a client and server in association with an application being accessed by the client at the server, comprising machine readable media containing one or more programs which when executed implement the steps of: configuring the server to store a model associated with the application and to execute view-generating and controller logic associated with the application; and configuring the client to store at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic associated with the application; wherein one or more portions of the application are performed at the client without the client having to interact with the server, and further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith.

In an illustrative embodiment of the recited article of manufacture (e.g., Specification, page 15, lines 10-13) for use in a client/server system (e.g., FIG. 3) for reducing interactions between a client and server in association with an application being accessed by the client at the server, comprising machine readable media containing one or more programs (e.g., Specification, page 15, lines 10-11. “software components including instructions or code for performing the methodologies of the invention”) which when executed implement the steps of: configuring the server (e.g., web

server 5000 in FIG. 3) to store a model (e.g., server-side model 5600 in FIG. 3) associated with the application and to execute view-generating (e.g., server-side view-generating logic 5400 in FIG. 3) and controller logic (e.g., server side-controller logic 5800 in FIG. 3) associated with the application; and configuring the client (e.g., web browser 1000 in FIG. 3) to store at least a subset of the model (e.g., client-side model 1600 in FIG. 3) associated with the application and to execute at least a subset of the view-generating (e.g., server-side view-generating logic 5400 in FIG. 3) and controller logic (e.g., server-side controller logic 5800 in FIG. 3) associated with the application; wherein one or more portions of the application are performed at the client without the client having to interact with the server (e.g., Specification, page 10, lines 16-18, “when a subset of the Model is stored on the client, as in accordance with the dual-MVC approach of the present invention, a number of interactions with the server can be eliminated”), and further wherein the client and the server both locally maintain at least a portion of the model (e.g., client-side model 1600 and server-side model 5600 in FIG. 3) and execute the view-generating (e.g., client-side view-generating logic 1400 and server-side view-generating logic 5400 in FIG. 3) and controller logic (e.g., client-side controller logic 1800 and server-side controller logic 5800) associated therewith.

Independent claim 26 recites a method for use in a client/server system of reducing interactions between a client and server in association with an application being accessed by the client at the server, the method comprising the steps of: configuring the server to: (i) store a model associated with the application; (ii) execute view-generating logic associated with the application; and (iii) execute controller logic associated with the application; and configuring the client to: (i) store at least a subset of the model associated with the application; (ii) execute at least a subset of the view-generating logic associated with the application; and (iii) execute at least a subset of the controller logic associated with the application; wherein one or more portions of the application are performed at the client without the client having to interact with the server, and further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith; further wherein, in accordance with such a dual model-view-controller arrangement, a model comprises application data, rules, and algorithms affecting the data, a view comprises a screen or window representation of a subset of the model that

the application chooses to display, and a controller comprises the logic that processes user requests, and causes the model to be changed and the view to be refreshed.

An illustrative embodiment of the recited method for use in a client/server system (e.g., FIG. 3) of reducing interactions between a client and server in association with an application being accessed by the client at the server, the method comprising the steps of: configuring the server (e.g., web server 5000 in FIG. 3) to: (i) store a model associated with the application (e.g., server-side model 5600 in FIG. 3); (ii) execute view-generating logic (e.g., server-side view-generating logic 5400 in FIG. 3) associated with the application; and (iii) execute controller logic (e.g., server-side controller logic 5800 in FIG. 3) associated with the application; and configuring the client (e.g., web browser 1000 in FIG. 3) to: (i) store at least a subset of the model (e.g., client-side model 1600 in FIG. 3) associated with the application; (ii) execute at least a subset of the view-generating logic (client-side view-generating logic 1400 in FIG. 3) associated with the application; and (iii) execute at least a subset of the controller logic (e.g., client-side controller logic 1800 in FIG. 3) associated with the application; wherein one or more portions of the application are performed at the client without the client having to interact with the server (e.g., Specification, page 10, lines 16-18, “when a subset of the Model is stored on the client, as in accordance with the dual-MVC approach of the present invention, a number of interactions with the server can be eliminated”), and further wherein the client and the server both locally maintain at least a portion of the model (e.g., client-side model 1600 and server-side model 5600 in FIG. 3) and execute the view-generating (e.g., client-side view-generating logic 1400 and server-side view-generating logic 5400 in FIG. 3) and controller logic (e.g., client-side controller logic 1800 and server-side controller logic 5800) associated therewith; further wherein, in accordance with such a dual model-view-controller arrangement, a model comprises application data, rules, and algorithms affecting the data (e.g., Specification, page 10, lines 18-20, “if the name, ID, phone operating system, operating system-specific information, and remarks are in the client’s model, i.e., are stored at the client”), a view comprises a screen or window representation (e.g., user interaction window 1200 in FIG. 3) of a subset of the model that the application chooses to display, and a controller comprises the logic that processes user requests (e.g., client-side controller logic 1800 in FIG. 3), and causes the model to be changed and the view to be

refreshed (e.g., Specification, page 10, lines 20-22, “(i) the screen can be refreshed with the addition of a blank row; and (ii) the screen can be refreshed by adding the proper operating system table without going to the server.

GROUND OF REJECTION TO BE REVIEWED ON APPEAL

(I) Whether claims 1, 3, 13, 15, 25 and 26 are anticipated under 35 U.S.C. §102(e) by U.S. Patent No. 6,161,136 (hereinafter “Hyndman”).

(II) Whether claims 2, 4-12, 14 and 16-24 are unpatentable under 35 U.S.C. §103(a) over Hyndman in view of U.S. Patent No. 5,768,510 (hereinafter “Gish”).

ARGUMENT

Appellants incorporate by reference herein the disclosure of their previous response filed in the present application, namely the response dated April 24, 2007.

(I) Whether claims 1, 3, 13, 15, 25 and 26 are anticipated under 35 U.S.C. §102(e) by U.S. Patent No. 6,161,136 (hereinafter “Hyndman”).

Regarding the §102(e) rejection of claims 1, 3, 13, 15, 25 and 26, the final Office Action contends that Hyndman discloses all of the claim limitations recited in the subject claims. Appellants respectfully assert that Hyndman fails to teach or suggest all of the limitations in claims 1, 3, 13, 15, 25 and 26, for at least the reasons presented below.

It is well-established law that a claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference. *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 U.S.P.Q.2d 1051, 1053 (Fed. Cir. 1987). Appellants assert that the rejection based on Hyndman does not meet this basic legal requirement, as will be explained below.

The claimed invention, as recited in independent claim 1, provides a method for use in a client/server system of reducing interactions between a client and server in association with an application being accessed by the client at the server. The method comprises the steps of:

configuring the server to store a model associated with the application and to execute view-generating and controller logic associated with the application; and configuring the client to store at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic associated with the application; wherein one or more portions of the application are performed at the client without the client having to interact with the server, and further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith. Independent claims 13, 25 and 26 recite similar limitations.

As explained on page 3, lines 22-27, of the present specification: “[t]he invention addresses performance by employing a dual-MVC approach, in which a subset of the application’s Model-View-Controller reside on the client, and the full Model-View-Controller and View-Generating-Logic reside on the server, thereby reducing the number of required server interactions.” FIG. 3 of the present application illustrates such an inventive dual-MVC approach.

Thus, the claimed invention recites that the server “store[s] a model associated with the application and maintain[s] view-generating and controller logic associated with the application,” and the client “store[s] at least a subset of the model associated with the application and maintain[s] at least a subset of the view-generating and controller logic associated with the application.”

While Hyndman discloses a multilevel model-view-controller (MMVC), Hyndman does not disclose the dual-MVC approach of the claimed invention. That is, among other deficiencies, Hyndman does not disclose having a server to store and maintain a model associated with the application and to execute view-generating and controller logic associated with the application, and having a client to store and maintain at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic associated with the application.

The Office Action relies on Hyndman at column 2, lines 40-49, which states the following with emphasis supplied:

According to a further aspect of the invention, there is provided a method of structuring functionality of a user interface for a network management system of a

communication network into a multilayered model-view-controller pattern, comprising separating the user interface into a server and a client layer integrating the state information on the server in a high-level model, integrating the state information on the client layer as a high-level view, and integrating a data marshalling mechanism provided by the network management system as a high-level controller.

The Office Action also relies on Hyndman at column 3 lines 59-64, which states the following with emphasis supplied:

At the high level, the two parts of the user interface according to the invention, namely the UIC and UIS, follow this MVC pattern. The UIS state information forms the model, denoted with M, and the data marshalling mechanism is the controller, denoted with C, and the view is the UIC portion of the interface, denoted with V.

FIG. 1B illustrates the next level of the MMVC architecture according to the invention. For example, MVC-1, comprising view V-1, model M-1 and controller C-1, is the view V-2 for the next level MVC-2. MVC-2, comprising view V-2, model M-2 and controller C-2, is the view V2 for the next level MVC-3. This continues until the finest granularity for any given system.

If the MMVC pattern is applied to the structure shown in FIG. 1B, at a high level, the view V-2 of the MVC-2 is the UIC state information, the model M-2 is the UIS and the controller C-2 is the data marshalling mechanism for the various applications provided in the MN.

The relied-upon portions of Hyndman do not teach or suggest having a server to store and maintain a model associated with the application and to execute view-generating and controller logic associated with the application, and having a client to store and maintain at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic associated with the application.

In response to Appellants' arguments, the Examiner refers to FIG. 1B and column 4, lines 1-25 of Hyndman as disclosing a server to store and maintain a model associated with the application and to execute view-generating and controller logic associated with the application, and disclosing a client to store and maintain at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic associated with the application.

In particular, the Examiner points to MVC-2 as teaching or suggesting the server layer MVC, and MVC-1 as teaching or suggesting the client layer MVC on page 2, fifth paragraph of the Final Office Action.

The Examiner appears to be suggesting that MVC-2 is a server storing and maintaining a model M-2 and executing view-generating with V-2 and controller logic with C-2, where M-2, V-2 and C-2 are associated with the application, and MVC-1 is a client storing and maintaining at least a subset of the model associated with the application (M-1) and executing at least a subset of the view generating with V-1 and controller logic with C-1 associated with the application. However, Hyndman at column 4, lines 6-8, states that “the view V-2 of MVC-2 is the UIC state information, the model M-2 is the UIS and the controller C-2 is the data marshalling mechanism for the various applications provided in the MN.” Therefore, V-2, as the UIC state information, does not teach or suggest the claimed view-generating logic that is associated with the application and executed by the server, and C-2, as the marshalling mechanism for the various applications provided in the MN, does not teach or suggest the claimed controller logic associated with the application and executed by the server. Furthermore, the relied-upon portion of Hyndman does not teach or suggest of a client storing and maintaining at least a subset of the model associated with the application and executing at least a subset of the view-generating and controller logic associated with the application. For example, in the relied-upon portion of Hyndman, the only reference to the client side controller logic, C-1, states that, “UIC comprises one or more mini-controllers C-1 that interact both with the model cache M-1 and the controllers C-2 on the server, to ensure that both models are kept in synchronization” (see Hyndman at column 4, lines 16-19). The relied-upon portion of Hyndman does not disclose the limitation of a client executing at least a subset of the controller logic associated with the application.

Thus, the architecture of Hyndman does not teach or suggest having a server to store and maintain a model associated with the application and to execute view-generating and controller logic associated with the application, and having a client to store and maintain at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic associated with the application.

Accordingly, Appellants assert that independent claims 1, 13, 25 and 26, as well as the claims which depend therefrom, are patentable over Hyndman and therefore allowable. Such dependent claims also recite patentable subject matter in their own right.

Further, Appellants assert that dependent claims 3 and 15 are patentable over Hyndman not only because they respectively depend from independent claims 1 and 13, but also because said claims recite patentable subject matter in their own right.

Accordingly, withdrawal of the §102(e) rejection is respectfully requested.

(II) Whether claims 2, 4-12, 14 and 16-24 are unpatentable under 35 U.S.C. §103(a) over Hyndman in view of U.S. Patent No. 5,768,510 (hereinafter “Gish”).

Regarding the §103(a) rejections to claims 2, 4-12, 14 and 16-24, Appellants respectfully assert that such dependent claims are patentable over the Hyndman/Gish combination for at least the reasons given above with respect to independent claims 1 and 13. Gish fails to remedy the deficiencies of Hyndman. However, Appellants also assert that such dependent claims also recite patentable subject matter in their own right.

Claim 4 recites “the configuring step further comprises the step of partitioning a screen area associated with the browser software into frames.” Claim 16 recites a similar limitation. The Examiner refers to column 35, lines 20-50 of Gish as teaching or suggesting the limitation. The relied-upon portion of Gish refers to creating a GUI and creating a frame to hold the GUI applet. Thus, Gish does not disclose partitioning a screen area associated with the browser software into frames.

Claims 5 and 17 recite “the at least a subset of the model, the view-generating and the controller logic associated with the application are associated with at least one frame and one or more views for display in accordance with the application are associated with at least another frame.” The Examiner refers to Gish at column 35, lines 20-50 and column 37, line 55 to column 38, line 40 as teaching or suggesting the limitation. As noted above, column 35, lines 20-50 of Gish refers to creating a frame to hold the GUI applet. Column 37, line 55 to column 38, line 40 of Gish refers to creating a handler that updates the user interface. No where does the relied-upon portions of Gish

teach or suggest the limitations of claims 5 and 17.

Claim 8 recites “the configuring step further comprises forming at least one frame with which application-independent view-generating logic and controller logic is associated.” Claim 20 recites a similar limitation. The Examiner refers to Gish at column 2, line 60 to column 3, line 45, column 35, lines 20-50 and column 45, line 55 to column 46, line 15 as teaching or suggesting the limitation. Column 2, line 60 to column 3, line 45 refers to window manager 218 maintaining all of the window displays that the user views during operation of the application programs. Column 35, lines 20-50 refers to creating a frame to hold the GUI applet. Column 45, line 55 to column 46, line 15 refers to a startup applet providing a way to launch an ICE-T application from a Web page. No where does the relied-upon portions of Gish teach or suggest the limitations of claim 8 and 20.

Claims 12 and 24 recite “the at least a subset of the model, the view-generating and the controller logic associated with the application are downloaded from the server to the client upon demand.” Gish does not teach or suggest the limitations of claims 12 and 24. For example, column 18, lines 14-67 of Gish refers to client node 500 contacting the server node 520 via HTTP with a request to execute an application, Gish does not teach or suggest that the at least a subset of the model, the view-generating and the controller logic associated with the application are downloaded from the server to the client upon demand.

Accordingly, withdrawal of the §103(a) rejections is respectfully requested.

Also, the Examiner has failed to identify a cogent motivation for combining Hyndman and Gish in the manner proposed. For example, for claims 2 and 14 the Examiner provides the following statement of motivation beginning at page 5, first paragraph of the Office Action:

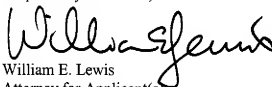
It would have been obvious for one of ordinary skill in the art to modify Hyndman wherein the client and server communicate over a HyperText Transport Protocol network as per the teachings of Gish because of the universal application of HTTP in Internet communication.

Appellants respectfully submit that this is a conclusory statement of the sort rejected by both the Federal Circuit and the U.S. Supreme Court. See KSR v. Teleflex, No. 13-1450, slip. op. at 14 (U.S., Apr. 30, 2007), quoting In re Kahn, 441 F. 3d 977, 988 (Fed. Cir. 2006) (“[R]ejections on

obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness.”). There has been no showing in the present §103(a) rejection of claims 2 and 14 of objective evidence of record that would motivate one skilled in the art to combine Hyndman and Gish to produce the particular limitations in question. The above-quoted statement of motivation provided by the Examiner appears to be a conclusory statement of the type ruled insufficient in KSR v. Teleflex.

In view of the above, Appellants believe that claims 1-26 are in condition for allowance, and respectfully request withdrawal of the §102 and §103 rejections.

Respectfully submitted,



William E. Lewis
Attorney for Applicant(s)
Reg. No. 39,274
Ryan, Mason & Lewis, LLP
90 Forest Avenue
Locust Valley, NY 11560
(516) 759-2946

Date: August 28, 2007

APPENDIX

1. A method for use in a client/server system of reducing interactions between a client and server in association with an application being accessed by the client at the server, the method comprising the steps of:

configuring the server to store a model associated with the application and to execute view-generating and controller logic associated with the application; and

configuring the client to store at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic associated with the application;

wherein one or more portions of the application are performed at the client without the client having to interact with the server, and further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith.

2. The method of claim 1, wherein the client and server communicate over a HyperText Transport Protocol network.

3. The method of claim 1, wherein the client performs the one or more portions of the application in accordance with browser software running thereon.

4. The method of claim 3, wherein the configuring step further comprises the step of partitioning a screen area associated with the browser software into frames.

5. The method of claim 4, wherein the at least a subset of the model, the view-generating and the controller logic associated with the application are associated with at least one frame and one or more views for display in accordance with the application are associated with at least another frame.

6. The method of claim 5, wherein the at least one view frame is a visible frame.

7. The method of claim 5, wherein the at least one frame associated with the at least a subset of the model, the view-generating logic and the controller logic is not a visible frame.

8. The method of claim 4, wherein the configuring step further comprises forming at least one frame with which application-independent view-generating logic and controller logic is associated.

9. The method of claim 8, wherein the at least one application-independent view-generating logic and controller logic frame further has an application-independent model associated therewith.

10. The method of claim 8, wherein the at least one application-independent view-generating logic and controller logic frame serves as an application programming interface for developing views to be displayed in accordance with the application.

11. The method of claim 10, wherein the views are implemented in accordance with the HyperText Markup Language and the application programming interface is implemented in accordance with the JavaScript language.

12. The method of claim 1, wherein the at least a subset of the model, the view-generating and the controller logic associated with the application are downloaded from the server to the client upon demand.

13. A network-based system, comprising:

a server having at least one processor operative to: (i) store a model associated with an application associated with the server; and (ii) execute view-generating and controller logic associated with the application; and

a client, coupled to the server via a network, having at least one processor operative to: (i)

store at least a subset of the model associated with the application; and (ii) execute at least a subset of the view-generating and controller logic associated with the application;

wherein one or more portions of the application are performed at the client without the client having to interact with the server such that interactions between the client and server are reduced, and further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith.

14. The system of claim 13, wherein the network is a HyperText Transport Protocol network.

15. The system of claim 13, wherein the client processor performs the one or more portions of the application in accordance with browser software running thereon.

16. The system of claim 15, wherein the client processor is further operative to partition a screen area associated with the browser software into frames.

17. The system of claim 16, wherein the at least a subset of the model, the view-generating and the controller logic associated with the application are associated with at least one frame and one or more views for display in accordance with the application are associated with at least another frame.

18. The system of claim 17, wherein the at least one view frame is a visible frame.

19. The system of claim 17, wherein the at least one frame associated with the at least a subset of the model, the view-generating logic and the controller logic is not a visible frame.

20. The system of claim 16, wherein the client processor is further operative to form at least one frame with which application-independent view-generating logic and controller logic is associated.

21. The system of claim 20, wherein the at least one application-independent view-generating logic and controller logic frame further has an application-independent model associated therewith.

22. The system of claim 20, wherein the at least one application-independent view-generating logic and controller logic frame serves as an application programming interface for developing views to be displayed in accordance with the application.

23. The system of claim 22, wherein the views are implemented in accordance with the HyperText Markup Language and the application programming interface is implemented in accordance with the JavaScript language.

24. The system of claim 13, wherein the at least a subset of the model, the view-generating and the controller logic associated with the application are downloaded from the server to the client upon demand.

25. An article of manufacture for use in a client/server system for reducing interactions between a client and server in association with an application being accessed by the client at the server, comprising machine readable media containing one or more programs which when executed implement the steps of:

configuring the server to store a model associated with the application and to execute view-generating and controller logic associated with the application; and

configuring the client to store at least a subset of the model associated with the application and to execute at least a subset of the view-generating and controller logic associated with the application;

wherein one or more portions of the application are performed at the client without the client having to interact with the server, and further wherein the client and the server both locally maintain

at least a portion of the model and execute the view-generating and controller logic associated therewith.

26. A method for use in a client/server system of reducing interactions between a client and server in association with an application being accessed by the client at the server, the method comprising the steps of:

configuring the server to: (i) store a model associated with the application; (ii) execute view-generating logic associated with the application; and (iii) execute controller logic associated with the application; and

configuring the client to: (i) store at least a subset of the model associated with the application; (ii) execute at least a subset of the view-generating logic associated with the application; and (iii) execute at least a subset of the controller logic associated with the application;

wherein one or more portions of the application are performed at the client without the client having to interact with the server, and further wherein the client and the server both locally maintain at least a portion of the model and execute the view-generating and controller logic associated therewith;

further wherein, in accordance with such a dual model-view-controller arrangement, a model comprises application data, rules, and algorithms affecting the data, a view comprises a screen or window representation of a subset of the model that the application chooses to display, and a controller comprises the logic that processes user requests, and causes the model to be changed and the view to be refreshed.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.